Санкт-Петербургский государственный университет
Филологический факультет
Кафедра математической лингвистики

Короткова Елизавета Алексеевна

АВТОМАТИЧЕСКОЕ ВЫДЕЛЕНИЕ НОВОСТНЫХ СООБЩЕНИЙ В СОЦИАЛЬНЫХ СЕТЯХ С ПОМОЩЬЮ ПРОГРАММЫ WEKA

Выпускная квалификационная работа по направлению
45.03.02 «Лингвистика»,
образовательная программа
«Прикладная, экспериментальная и математическая лингвистика»

Научный руководитель:
доц., к.ф.н.
Митренина О.В.

Санкт-Петербург
2017

Оглавление

Введение

Машинное обучение как область знаний занимается разработкой, оценкой и исследованием алгоритмов, способных выявлять скрытые закономерности в больших массивах данных. Методы машинного обучения делают возможной автоматическую классификацию объектов на основе их признаковых описаний. По мере накопления объема анализируемых данных качество классификации улучшается.

Технологии машинного обучения широко используются в различных целях, к примеру, в маркетинговых исследованиях, в медицине, в робототехнике. Их применение возможно в тех областях, где необходима эффективная автоматическая обработка больших массивов эмпирических данных.

Машинное обучение все чаще используется и для решения задач компьютерной лингвистики. В круг таких задач входят машинный перевод, определение тональности (эмоциональной окраски) текстов, морфологический анализ, автоматическая рубрикация текстов, распознавание речи.

В данной работе исследуется возможность использования алгоритмов машинного обучения для автоматического выделения новостных текстов в социальных сетях. С этой целью анализируется массив текстов на русском языке, для которого создается признаковое описание. К полученному массиву описанных по выбранной модели текстов применяется алгоритм машинного обучения. Проверка результатов построения модели по данным проводится с помощью массива ранее не анализировавшихся данных. Результаты работы модели на этих данных сравниваются с информацией об истинной принадлежности текстов классу новостных или личных сообщений. Этот процесс повторяется несколько раз с использованием различных моделей признакового описания текстов, что позволяет выявить лучший набор признаков для классификации сообщений.

Цель работы – сравнительный анализ различных моделей формального признакового описания русскоязычных текстов для выделения новостных сообщений в социальных сетях из общего массива сообщений.

Для достижения этой цели выполняются несколько задач:

- изучение принципов машинного обучения;
- сбор данных, создание тестовой и обучающей выборки;
- разработка алгоритма классификации:
    - настройка ПО Weka для работы с русскоязычными текстами,
    - описание текстов при помощи различных наборов признаков,
    - преобразование данных в нужный для классификации в Weka формат,
    - обучение модели,
    - классификация контрольной выборки при помощи построенной модели,
- оценка результатов классификации.

Материал для исследования – случайная выборка из 1348 коротких текстов на русском языке. Тексты были собраны в социальной сети Twitter при помощи программы Webometric Analyst и затем обработаны. В наборы признаков, использованные на разных этапах работы, вошли векторные представления нелемматизированных и лемматизированных текстов, упоминания автором твита профилей других пользователей социальной сети, хэштеги, ссылки в тексте сообщения, а также наличие в сообщении глаголов и имен собственных.

Использованный алгоритм машинного обучения – наивный байесовский классификатор. Он был выбран как достаточно простой, но в то же время эффективный для признаковых пространств большой размерности способ построения обучаемой модели.

Актуальность работы связана с интенсивным развитием использования методов машинного обучения в автоматической обработке текстов, в том числе текстов из социальных сетей, которые становятся все более значимым источником информации о происходящих в мире событиях.

Практическая значимость проделанной работы обусловлена возможностью дальнейшего использования разработанных алгоритмов определения признаков твитов, а также опыта применения ПО Weka для классификации русскоязычных

текстов. Полученные модели могут использоваться в качестве первой ступени агрегации новостей в социальной сети Twitter.

Глава 1. Машинное обучение в прикладной лингвистике

1.1. Общие принципы машинного обучения

Задачу машинного обучения (machine learning) в ее типичном виде можно описать следующим образом. Имеется некоторое множество объектов, каждому из элементов которого может быть приписан признак из множества, именуемого множеством ответов. Систему, по которой объекту приписывается ответ, называют целевой функцией (target function). Задача машинного обучения – подобрать такую функцию, которая, с одной стороны, наиболее близка к целевой, то есть почти всегда дает «правильные» ответы для заранее не известных объектов, с другой стороны, реализуема на компьютере, универсальна и не требует значительных человеческий усилий для реализации. Функция, имитирующая целевую, подбирается из некоторого ограниченного множества. Выбор этой функции и подбор её параметров и осуществляется одним из алгоритмов машинного обучения.

Следующим этапом после подбора аппроксимирующей функции является оценка ее близости к целевой функции. Поскольку проверить работу функции на всем множестве потенциальных объектов невозможно, проводят статистическую оценку на контрольной выборке.

Приближение целевой функции может строиться по заранее известным данным. Такой метод называют обучением с учителем (supervised learning), а данные – обучающей выборкой (training set). [Протопопова, Букия 2016] В этом случае целевая функция промерена в конечном множестве точек, называемом обучающей выборкой. Таким образом, обучающая выборка – множество пар объект-ответ, по которому требуется построить функцию, аппроксимирующую неизвестную зависимость. Иногда обучающая выборка появляется постепенно, корректируя алгоритм. Например, некоторые автоматические переводчики позволяют пользователям выбрать наиболее удачный перевод и корректируют свою работу, основываясь на полученных данных. Третья разновидность алгоритмов – обучение без учителя (unsupervised learning). Такие алгоритмы, например, разбивают

объекты на группы, называемые кластерами, причем в одном кластере оказываются близкие объекты. Впоследствии всем элементам кластера присваивается один и тот же ответ. Этот способ практически не требует заранее обработанных данных.

Самый распространённый способ задания объектов — признаковое описание. С формальной точки зрения признаки — это функции, которые ставят в соответствие объектам какие-либо значения. На содержательном уровне признаки – это некоторые измерения над объектами. Признаки могут быть следующих типов:

- бинарные (каждому объекту соответствует ответ «да/нет»);
- номинальные (принимают большее, чем бинарные признаки, но конечное число значений);
- порядковые (на множестве значений признаков задано отношение порядка);
- количественные (числовые измерения над объектами).

Во многих задачах разнотипные признаки оказываются смешаны.

Таким образом, обучающую выборку можно представить в виде матрицы, строки которой соответствуют отдельным объектам, а столбцы – признакам, и каждой строке соответствует некоторый правильный ответ.

Множество ответов может состоять из двух элементов (задачи, в которых требуется принять одно из двух возможных решений, например, определение пола автора текста), из нескольких элементов (задачи классификации, например, рубрикация документов по темам), быть множеством естественных чисел (задачи восстановления регрессии, например, определение времени создания текста). Также ответом может быть упорядоченное множество объектов (задачи ранжирования, которые решаются, в частности, поисковыми системами при ранжировании поисковой выдачи). [Воронцов 2007]

Общее определение машинного обучения сформулировано Т. Митчеллом следующим образом: «Компьютерная программа обучается на опыте E по отношению к некоторому классу задач T и мере качества P, если качество решения задач из класса T этой программой, измеренное мерой P, улучшается с приобрете-

нием опыта E». Основное предположение при решении задачи методами машинного обучения с учителем состоит в том, что решающая функция, хорошо приближающая целевую функцию на достаточно большой обучающей выборке, будет хорошо приближать целевую функцию на всем множестве объектов. [Mitchell 1997]

1.2. Наивный байесовский классификатор

Среди методов обучения с учителем для классификации объектов наиболее распространены байесовские методы классификации (Bayesian Learning), машины опорных векторов (Support Vector Machines) или метод опорных векторов, использование деревьев принятия решений (Decision-Tree Learning).

В данной работе использовался наивный байесовский классификатор.

В байесовских методах классификации вычисляются вероятности принадлежности объекта к тому или иному классу, при этом каждый объект выборки влияет только на вероятность принятия решения о классификации новых объектов, но не приводит непосредственно к принятию положительного или отрицательного решения. Вычисления, производимые этими классификаторами, основаны на теореме Байеса:

$$P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)},$$

где $P(A|B)$ — апостериорная вероятность события (гипотезы) A при наступлении события B;

$P(B|A)$ — апостериорная вероятность наступления события B при истинности гипотезы A;

$P(A)$, $P(B)$ — априорные вероятности событий A и B.

Работа вероятностного классификатора заключается в вычислении значений $P(c_i|\vec{d}_j)$ (вероятность того, что документ, представленный признаковым вектором $\overrightarrow{d_J}$, относится к классу c_i) для каждого класса c_i из множества всех классов C и нахождении наибольшей такой вероятности для каждого нового объекта:

$$H(\vec{d}_j) = \underset{c_i \in C}{argmax}\, P(c_i|\vec{d}_j)$$

Условную вероятность $P(c_i|\vec{d}_j)$ можно переписать по теореме Байеса как

$$P(c_i|\vec{d}_j) = \frac{P(\vec{d}_j|c_i)\cdot P(c_i)}{P(\vec{d}_j)},$$

где $P(c_i)$ – априорная вероятность отнесения документа к классу c_i, $P(\vec{d}_j|c_i)$ – вероятность обнаружить документ, представленный вектором $\vec{d}_j$, в классе c_i, $P(\vec{d}_j)$ – вероятность того, что случайный документ будет иметь вектор $\vec{d}_j$. [Епрев 2010]

Как правило, количество потенциально существующих признаковых векторов очень велико из-за их большой размерности (в частности, в задачах классификации текстов, где количество признаков может быть очень большим), что затрудняет построение решающей функции. Чтобы решить эту проблему, при построении модели наивного байесовского классификатора для вычисления $P(\vec{d}_j|c_i)$ и $P(\vec{d}_j)$ делается предположение, что любые две координаты вектора документа, т.е. любые два признака документа, статистически независимы друг от друга. [Sebastiani 2002] Это предположение, вообще говоря, в большинстве случаев неверно. Однако, несмотря на простоту метода, он удобен в задачах с большой размерностью признакового пространства и часто показывает лучшие результаты, чем более сложные альтернативы. [Hastie et al. 2009] [Rish 2001]

1.3. Использование машинного обучения для автоматического анализа сообщений

Классификация текстовых документов при помощи машинного обучения применяется, например, для сужения области поиска в поисковых системах, фильтрации спама, настройки контекстной рекламы, снятия омонимии при автоматическом переводе. [Епрев 2010]

Для анализа сообщений в социальных сетях методы машинного обучения стали использоваться относительно недавно, но в настоящее время эта область

исследований активно развивается.

Для оценки популярности и продвижения продуктов и услуг в социальных сетях используется анализ тональности сообщений и отзывов. В работе [Melville et al. 2009] для определения тональности записей в блогах, в которых отражено мнение пользователей об определенных компаниях и их продуктах, использовался комбинированный подход: вместе с традиционным использованием заранее составленного словаря эмоционально окрашенной лексики применялась автоматическая классификация полиномиальным наивным байесовским алгоритмом (эта модель предполагает появление каждого слова в документе независимо от других слов). Рассматривались тексты, содержащие оценку программного обеспечения IBM Lotus. При одновременном использовании двух подходов были получены лучшие результаты, чем при использовании каждого метода классификации по отдельности.

В статье [Wan, Gao 2015] алгоритмы машинного обучения также использовались для автоматического определения тональности сообщений. Выборка из 12864 твитов, в которых авторы выражали свое мнение об авиакомпаниях, была использована для обучения и тестирования семи различных моделей: одного классификатора, использующего информацию об эмоционально окрашенной лексике, и шести классификаторов, использующих принципы машинного обучения, в том числе наивного байесовского классификатора. При применении классификаторов по отдельности наивный байесовский классификатор стабильно показывал один из лучших по точности результатов. Итоговый класс твита определялся большинством голосов различных классификаторов. Такой подход позволил увеличить эффективность классификации.

Помимо определения тональности отзывов, при помощи машинного обучения выявляют также отзывы, которые являются спамом. [Jindal, Liu 2008]

Лингвистические признаки, извлекаемые из твитов, использовались также в работе [Pennacchiotti, Popescu 2011] для классификации профилей пользователей социальной сети Twitter в зависимости от их политических взглядов, этнической принадлежности и отношению к определенной компании. Для принятия решений

использовалась модификация комитета решающих деревьев (случайного леса). В число лингвистических признаков входили типичные для класса пользователей слова, типичные хэштеги, упоминания пользователей, упоминания которых характерны для класса, также использовалось тематическое моделирование (латентное размещение Дирихле). В статье показана значимость таких лингвистических признаков для классификации пользователей на основе твитов.

В [Lai et al. 2016] по текстам твитов определялось отношение пользователей к кандидатам в президенты на выборах в США в 2016 г. В качестве модели был выбран наивный байесовский классификатор, в качестве признаков – как структурные (хэштеги, упоминания), так и лексические данные.

В работе [Mukherjee, Liu 2010] описывается метод определения пола авторов записей в блогах с использованием в качестве признаков последовательностей частеречных комбинаций переменной длины.

В исследовании [Kunneman, van den Bosch 2014] методы машинного обучения использовались для выделения не известных заранее значимых событий по сообщениям в Twitter. Выборка состояла из 65 млн твитов на нидерландском языке, опубликованных в течение двух месяцев. Для кластеризации использовался метод k ближайших соседей.

Наиболее близкое к задачам данной работы исследование представлено в [Vikre, Wold 2015]. Авторами были поставлены задачи выделения новостных твитов, т.е. твитов, относящихся к затрагивающим многих людей событиям (в отличие от твитов личного характера), кластеризации этих твитов по отнесенности к определенному событию и выбора из каждого кластера твита, наилучшим образом отражающего описываемое событие. Материалом послужил корпус, состоящий из твитов на английском языке. На первом этапе для выделения и кластеризации новостных твитов использовались методы тематического моделирования. При сравнении результатов распределения твитов по темам с темами новостей из корпуса газеты New York Times алгоритм показал низкую точность. На втором этапе исследования для выделения и кластеризации новостей использовался метод нахождения ближайшего к заданному тексту в векторном пространстве доку-

мента (locality-sensitive hashing) вместе с разметкой именованных сущностей с использованием алгоритмов машинного обучения. При этом точность выделения релевантных кластеров составила 0,917.

Проблема автоматической классификации текстов исследовалась и на материале русского языка. В статье [Котельников, Клековкина 2012] рассматривается задача анализа тональности с использованием методов машинного обучения. Обсуждаются вопросы выбора оптимального варианта векторной модели представления текстов и наиболее подходящего алгоритма машинного обучения, в том числе рассматриваются наивный байесовский классификатор и метод опорных векторов.

В работе [Васильев, Худякова, Давыдов 2012] методами машинного обучения также анализируются тексты отзывов с целью их классификации в соответствии с тональностью. Однако, после анализа таких алгоритмов, как деревья принятия решений, метод опорных векторов и метод ближайших соседей, из которых наиболее удачным в рамках задачи оказался метод опорных векторов, авторы приходят к выводу, что наиболее эффективен подход, основанный на ручном построении правил экспертами.

В статье [Романов, Мещеряков 2011] авторы использовали метод опорных векторов для определения пола авторов коротких сообщений длиной от 20 до 200 символов. К наиболее существенным характеристикам были отнесены употребление определенных сочетаний букв, служебных слов, знаков пунктуации, придание эмоциональной окраски высказыванию с помощью эмотиконов, длина слов в сообщениях. Авторам удалось достигнуть максимальной точности классификации 74% при одновременном использовании различных признаков.

В статье [Адаскина, Паничева, Попов 2015] описано использование машинного обучения для анализа тональности твитов на русском языке. Из твитов извлекалось мнение пользователей о банках и телекоммуникационных компаниях. Наилучший результат был достигнут с использованием метода опорных векторов, самыми значимыми признаками оказались леммы, биграммы и синтаксические связи между словами.

1.4. Выводы к главе 1

В этой главе мы рассмотрели машинное обучение как раздел прикладной лингвистики, а также описали некоторые эксперименты по применению машинного обучения для автоматического анализа сообщений.

Для решения разных задач используются различные модели обучения и признаки объектов, при этом часто наибольшую эффективность показывают относительно простые модели. В эксперименте, проведенном в данной работе, использовался один из таких несложных алгоритмов – наивный байесовский классификатор. Он учитывает каждый признак объекта независимо и эффективен на признаковых пространствах с высокой размерностью.

Глава 2. Автоматическое выделение новостных сообщений в социальной сети Twitter

2.1. Общее описание эксперимента

В данной главе описывается эксперимент по классификации текстов из социальной сети Twitter с целью автоматического выделения из общего массива твитов новостных сообщений. Многие новостные агентства и издания публикуют в Twitter новости в сжатой форме, часто со ссылкой на заметку или статью о событии. Примеры таких сообщений:

- "Восемнадцать тонн мандаринов украли в Петербурге https://t.co/0dqFOqsdxS"
- "Во Франции задержаны подозреваемые в пособничестве террористам #Франция #спецслужбы #теракты #новости #news https://t.co/MhHl0pkb5q"
- "Петербурженку задержали за попытку заказать киллеру своих компаньонов https://t.co/VJVhSB7m2T"
- "Сноубордист, которого неделю искали в горах, прятался от семьи в Волгограде https://t.co/kGYKfhf1We https://t.co/F2a0LMwGiT"

Все тексты, не являющиеся новостными сообщениями, считались сообщениями «личного характера». Подборка примеров новостных и личных текстов представлена в приложениях 1 и 2.

Анализировались результаты работы наивного байесовского классификатора. Для описания текстов применялись пять различных наборов признаков. На первом этапе учитывались структурные признаки твита: количество в нем хэштегов, упоминаний автором твита других пользователей и наличие ссылок на веб-страницы. Затем использовалась морфологическая информация: наличие в тексте глаголов и имен собственных. На третьем этапе эти признаки были использованы одновременно. Затем для классификации применялись векторные представления текстов и, наконец, на последнем этапе были объединены все признаки.

Эксперимент проводился на корпусе русскоязычных сообщений из социальной сети Twitter (твитов). Длина твитов не превышает 140 символов.

Тексты были собраны при помощи программы Webometric Analyst. Отдельно были собраны сообщения, не являющиеся новостными, и новости. Из корпуса были удалены не подходящие для анализа тексты, затем оставшиеся тексты были помечены как личные или новостные. Корпус был разделен на обучающую (для обучения классификатора) и тестовую (для оценки результатов классификации) выборки. Общий объем корпуса составил 1348 сообщений.

Критериями оценки качества служили полнота и точность классификации по каждому из двух классов, а также F-мера по каждому классу и количество верно и неверно классифицированных объектов (суммарное и в каждом классе).

2.2. Создание обучающей и тестовой выборки

2.2.1. Сбор данных

Для проведения эксперимента были собраны две выборки сообщений в социальной сети Twitter.

Для создания выборок использовалась программа Webometric Analyst 2.0 [Thelwall 2009]. Это программное обеспечение предназначено для автоматического сбора информации в сети Интернет для проведения социологических исследований. Оно было разработано в Университете Вулвергемптона в Великобритании и распространяется свободно. Webometric Analyst позволяет, в частности, загружать веб-страницы и комментарии из социальных сетей (Twitter, Flickr, YouTube). Программа автоматически отправляет поисковые запросы на внешнюю информационно-поисковую систему через интерфейс прикладных программ (API), позволяя собрать большие объемы текстовых данных и метаданных, связанных с ними.

В данной работе использовалась функция сбора сообщений из социальной сети Twitter – твитов. В первую выборку вошли случайные твиты, в качестве единственного ограничения был установлен язык, на котором написаны сообщения – русский (Webometric Analyst позволяет собирать сообщения на определённом языке). После предобработки, описанной ниже, собранные случайные твиты образовали выборку, состоящую из сообщений личного характера. Объем этой выборки составил 898 сообщений.

Вторая группа – новостные сообщения – была собрана со страниц новостных изданий в социальной сети Twitter. Для этого использовалась функция загрузки сообщений, опубликованных определёнными пользователями за некоторый промежуток времени. Список пользователей загружается в Webometric Analyst в виде текстового файла. В него вошли Twitter-аккаунты информационных агентств (ИТАР-ТАСС, РИА Новости), федеральных телевизионных каналов (Первый канал, Пятый канал), новостных интернет-изданий (Lenta.ru), региональных новостных интернет-изданий (Фонтанка.ру), газет (Российская газета). Всего были собраны сообщения из 13 источников. Из массива была получена выборка новостных сообщений объемом 450 текстов.

Помимо текстов сообщений программа собирала также метаданные: дату и время публикации твитов, названия профилей опубликовавших их пользователей. Тексты сообщений и метаданные были сохранены программой Webometric Analyst в виде текстового файла, в котором поля разделены знаками табуляции.

2.2.2. Предварительная обработка данных

При анализе использовались только тексты твитов, дополнительная метаинформация не учитывалась. Первый этап предварительной обработки происходил с использованием программы для работы с электронными таблицами MS Excel 2013 и текстового редактора Notepad++.

На этапе предварительной обработки из первой группы твитов, собранной только с ограничением на язык, были вручную удалены по ошибке попавшие в неё твиты, не содержащие текста на русском языке, например, содержащие только ссылку на веб-страницу или текст на ином языке. Были также удалены очевидно автоматически сгенерированные сообщения и поисковые запросы. В итоге был получен массив из 898 твитов личного характера.

Из группы новостных сообщений было отобрано 450 твитов. В их число вошли сообщения, затрагивающие различные темы и опубликованные источниками разных форматов, были удалены повторяющиеся тексты.

Из отобранных сообщений была сформирована обучающая выборка из 899

объектов (599 сообщений личного характера и 300 новостей) и тестовая выборка из 449 объектов (299 сообщений личного характера и 150 новостей). Данные были сохранены в виде таблицы, содержащей два столбца. В первый столбец записывался текст твита, во второй – метка класса (1 для личных сообщений, 2 – для новостей).

Для работы в программе Weka данные были сохранены в формате .csv (Comma-Separated Values) – текстовом формате, предназначенном для представления табличных данных, где каждая строка файла соответствует строке таблицы, а разделителями значений столбцов служат запятые. При экспорте данных формата .xlsx в формат .csv при помощи ПО MS Excel в качестве разделителя используется точка с запятой. Разделители были заменены на запятые при помощи замены регулярных выражений. Для правильного распознавания данных программой Weka при помощи поиска и замены по регулярным выражениям в текстовом редакторе Notepad++ текст каждого твита был заключен в кавычки. Все кавычки, содержавшиеся внутри текстов, были удалены, так как вызывали ошибки при распознавании программой Weka границ полей. Поскольку ПО Weka с изначальной конфигурацией настроек не распознаёт кириллические символы, понадобилось изменить настройку кодировки в файле RunWeka.ini в установочной директории программы на UTF-8 и сохранить все файлы в той же кодировке.

2.3. Обработка данных с помощью программы Weka

Файлы были загружены для обработки в Weka с помощью вкладки Preprocess интерфейса Explorer. При открытии файла в формате .csv каждая строка файла распознается как объект, каждое поле строки соответствует определенному признаку объекта. Названия признаков считываются из первой строки файла.

Некоторые типы данных изначально распознаются программой неправильно. Их тип можно изменить с помощью фильтров. К примеру, метки классов, в качестве которых было использовано число 1 для твитов личного характера и 2 для новостей, были распознаны как числовые данные. Для преобразования их в номинальный тип был применен фильтр

weka.filters.unsupervised.attribute.StringToNominal.

Для преобразования других типов применяются аналогичные фильтры.

После приведения всех данных к нужным типам файлы с исходными данными обучающей и тестовой выборки сохраняются для дальнейшей работы в формате .arff (Attribute Relation File Format). В файле .arff хранятся данные о признаках, их типах и примененных фильтрах, а также признаковая информация об объектах.

Построение модели классификации происходит на вкладке Classify. Для построения модели необходимо выбрать классификатор, способ проверки результативности модели и название столбца таблицы исходных данных, в котором записаны метки классов, присвоенных объектам экспертом при разметке. Оценка результатов возможна на той же выборке, на которой производится обучение, на отдельной тестовой выборке, путем перекрестной проверки (поочередного выбора разных обучающей и тестовой выборки из одного массива данных заданное количество раз) или одноразового выделения из массива данных тестовой выборки определенного объема. В данной работе использовалась опция “Supplied test set” – загрузка тестовой выборки в виде отдельного от обучающей выборки файла.

По результатам работы построенного классификатора программа выдает различные численные показатели результативности модели (процент верно классифицированных объектов, точность, полнота, F-мера, матрица ошибок и др.).

Полнота по классу равна отношению количества верно классифицированных объектов класса к числу всех элементов класса. Точность – отношение верно классифицированных объектов класса к общему числу объектов, отнесенных классификатором к данному классу.

2.4. Анализ с помощью структурных признаков

На первом этапе в качестве признаков были использованы показатели, легко выявляемые из строкового представления твита: количество хэштегов (меток, которые используются в социальных сетях и блогах для распределения сообщений по темам; могут включать буквы, цифры и знаки нижнего подчеркивания, последовательности этих символов предшествует знак #) и упоминаний пользователей

(последовательностей букв латинского алфавита, цифр и знаков нижнего подчеркивания, которым предшествует знак @) в тексте твита, а также наличие ссылок на веб-страницы. Эти признаки были выбраны, поскольку очевидно различие их распределений для разных классов (см. рисунки 1, 2, 3).

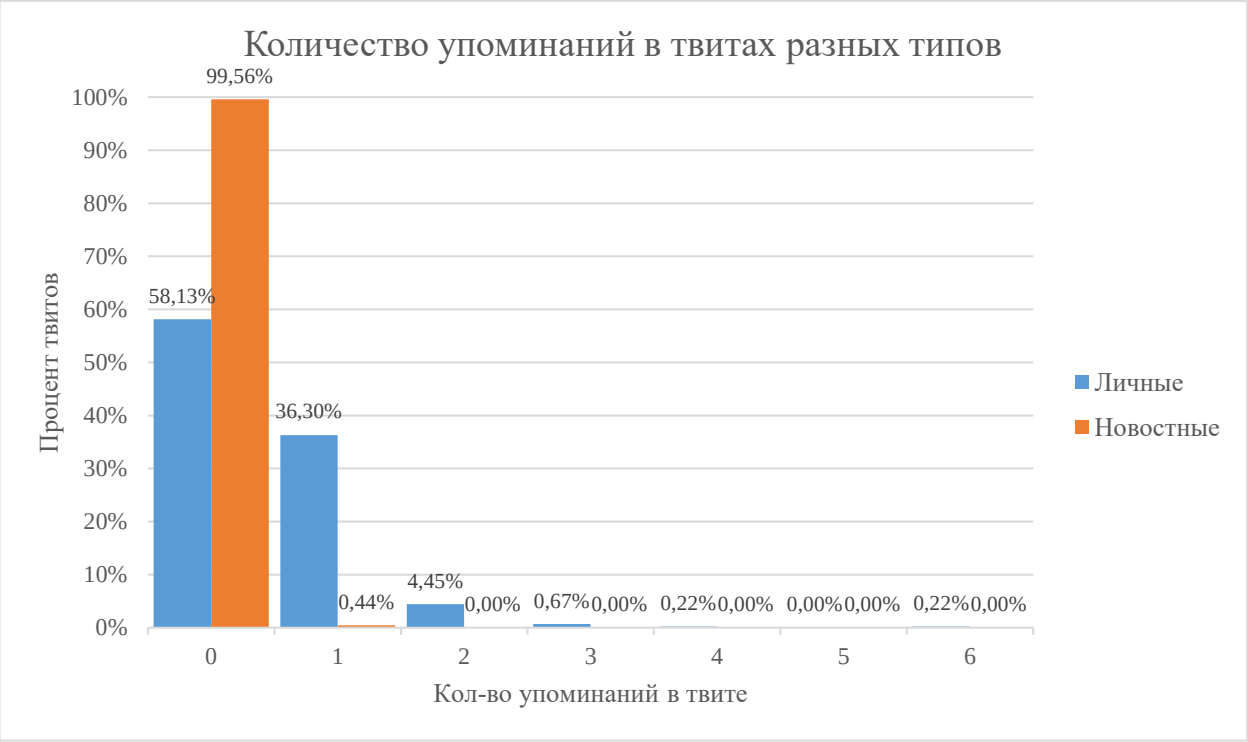

Рисунок 1. Количество упоминаний пользователей в твитах личного характера и новостных твитах

Ссылки на веб-страницы не встречаются лишь в 3 новостных текстах, вошедших в выборку, и присутствуют в подавляющем большинстве новостей в Twitter, поскольку новостные твиты, как правило, представляют собой начало или сжатую версию более длинного новостного сообщения и завершаются ссылкой на веб-страницу с полным текстом новости. Упоминания других пользователей, напротив, встречаются преимущественно в твитах личного характера. Хэштеги, как правило, также более характерны для личных твитов, но отдельные новостные аккаунты, к примеру, Пятый канал (@5tv), в большом количестве используют хэштеги в своих записях.

Нужно отметить, что с 30 марта 2017 г. упоминания пользователей в Twitter выносятся за пределы текста твита. Выборка была собрана до этого нововведения. При анализе сообщений, опубликованных после 30 марта 2017 г., придется либо

отказаться от использования информации об упоминаниях пользователей, либо извлекать ее не из текста сообщения, а из метаданных.

Рисунок 2. Количество хэштегов в твитах личного характера и новостных твитах

Для выделения признаков был реализован алгоритм на языке программирования Python в версии 3.5 (см. приложение 3). При написании программного кода использовалась интегрированная среда разработки Spyder. Алгоритм считывает файл в формате .csv, в который записаны тексты твитов и метки классов, к которым эти твиты отнесены. При помощи модуля re стандартной библиотеки Python, предназначенного для обработки текста с поддержкой синтаксиса регулярных выражений, программа распознает упоминания, хэштеги и ссылки в текстах твитов. На выходе создаётся новый файл в формате .csv, в котором под соответствующими заголовками столбцов “Mentions”, “Hashtags”, “Links” и “Label” записано количество упоминаний пользователей в твите, количество хэштегов, наличие либо отсутствие ссылок (1 или 0) и полученная из входного файла метка класса.

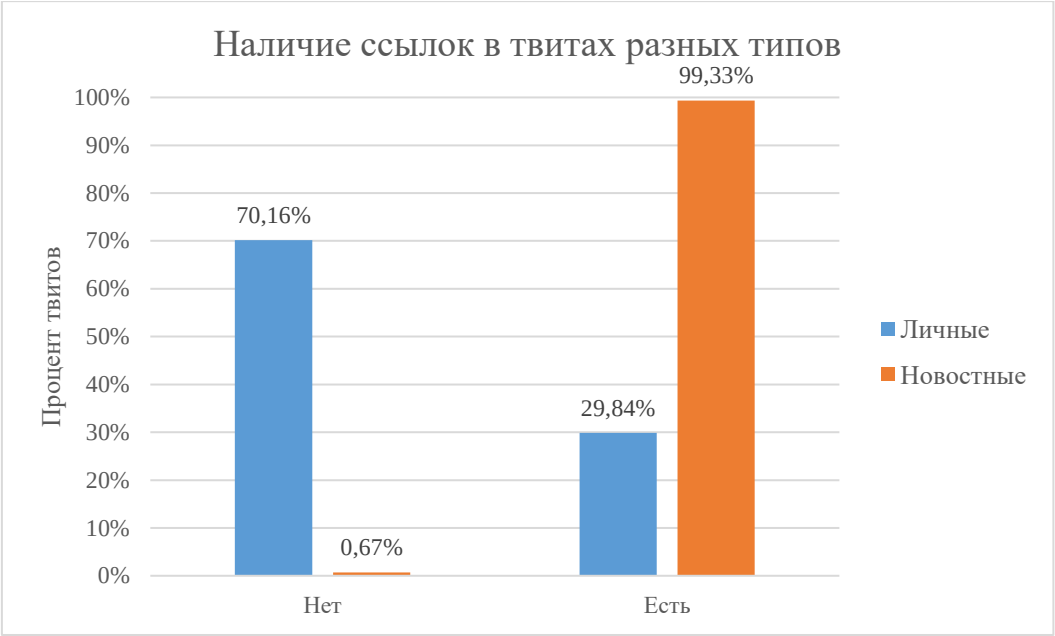

Рисунок 3. Наличие ссылок в твитах личного характера и новостных твитах

При использовании этих признаков для обучения наивного байесовского классификатора было верно классифицировано 311 из 449 объектов тестовой выборки, или 69,3% объектов. При этом большинство ошибок было вызвано ошибочной классификацией новостных твитов как личных. Полнота распознавания новостей составила 0,213. (См. значения полноты и точности в таблице 1 и матрицу ошибок в таблице 2).

Таблица 1. Точность и полнота классификации при использовании в качестве признаков информации об упоминаниях, хэштегах и ссылках в тексте

	Точность	Полнота
Личные сообщения	0,703	0,933
Новости	0,615	0,213

Таблица 2. Матрица ошибок классификатора при использовании в качестве признаков ссылок, хэштегов и упоминаний

Истинный класс	Классифицированы как	
	Личные сообщения	Новостные сообщения
Личные сообщения	279	20
Новостные сообщения	118	32

2.5. Анализ с помощью морфологической информации

На втором этапе признаками твитов служила информация о наличии в них глаголов и имен собственных. Поскольку новостные сообщения передают в максимально сжатом виде информацию о конкретных событиях и часто имеют схожую структуру, в них регулярно встречаются глаголы, имена и фамилии людей, а также географические названия.

Для получения грамматической информации использовался морфологический анализатор MyStem, разработанный в компании Яндекс, в оболочке для языка Python pymystem3.

MyStem – морфологический анализатор для русского языка, поддерживающий снятие морфологической неоднозначности, разработанный Ильёй Сегаловичем и Виталием Титовым в компании «Яндекс». Программа работает на основе словаря и способна формировать морфологические гипотезы о незнакомых словах. [Segalovich 2003] Лицензии разрешают свободное использование программы MyStem и оболочки pymystem3 в некоммерческих и коммерческих целях.

Для автоматического извлечения из выдачи морфологического анализатора информации о наличии глаголов и имен собственных в текстах также был реализован алгоритм на языке программирования Python 3 (см. приложение 4). Как содержащие глаголы были обозначены все тексты, в которых как минимум один токен был распознан морфологическим анализатором MyStem как глагол. Именами собственными считались токены, в грамматическое описание которых входят метки «гео» (географическое название), «имя» (имя собственное) или «фам» (фамилия) [Расшифровка граммем]. При этом из результатов исключались токены, в грамматическом описании которых имеется помета «“qual” = “bastard”». Это слова, которых нет в словаре морфологического анализатора MyStem, оцененные им как с большой вероятностью содержащие ошибку. [Rakhilina et al. 2016] Поскольку для сообщений личного характера в социальной сети Twitter характерна ненормативность и использование неологизмов, количество таких словоформ в выборке велико, и многие такие словоформы MyStem классифицирует как имена

или фамилии. Кроме этого, нужно заметить, что многие иностранные фамилии (например, Памук, Розенстайн) и топонимы местной значимости (Белоостров, Хемниц) не распознаются программой как таковые. По этим причинам полученное признаковое описание не полностью отражает реальное наличие имен собственных, а в некоторых случаях и глаголов в текстах.

На рисунках 4 и 5 показано различие распределений признаков для двух классов сообщений. При подсчете учитывалось не реальное наличие глаголов и имен собственных в текстах, а результаты автоматической разметки, так как именно эти результаты используются в дальнейшем для классификации.

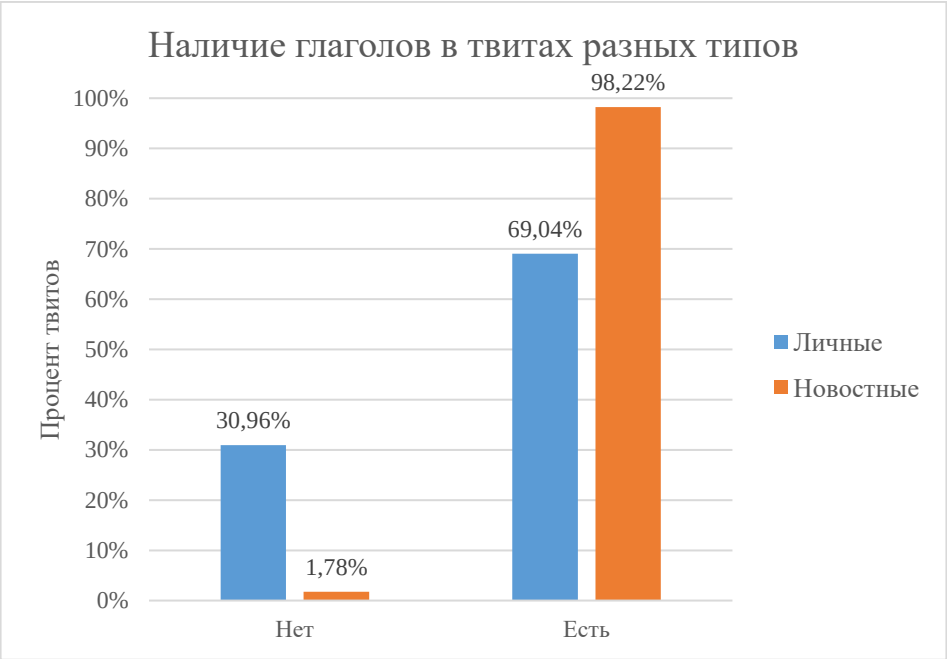

Рисунок 4. Наличие глаголов в личных и новостных твитах

Глаголы присутствуют в 78,8% всех твитов выборки, но особенно преобладают в новостных текстах, где они встречаются в 98,2% твитов. Имена собственные же были выделены в 71,1% новостных твитов, но лишь в 16,8% личных.

При использовании этих признаков было верно классифицировано 356 из 449 объектов, или 79,3% тестовой выборки. Морфологическая информация позволила повысить полноту распознавания новостей до 0,647. В таблицах 3 и 4 указаны полнота и точность классификации для двух типов сообщений и матрица ошибок.

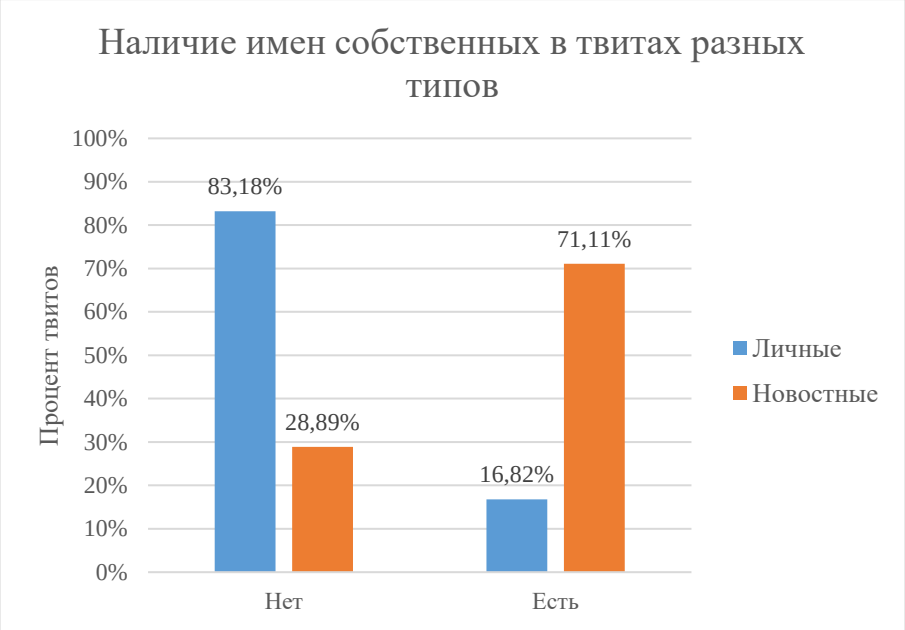

Рисунок 5. Наличие имен собственных в личных и новостных твитах

Таблица 3. Точность и полнота классификации при использовании в качестве признаков информации о наличии глаголов и имен собственных в тексте

	Точность	Полнота
Личные сообщения	0,830	0,866
Новости	0,708	0,647

Таблица 4. Матрица ошибок классификатора при использовании в качестве признаков информации о наличии глаголов и имен собственных в тексте

Истинный класс	Классифицированы как	
	Личные сообщения	Новостные сообщения
Личные сообщения	259	40
Новостные сообщения	53	97

2.6. Анализ с помощью структурных признаков и морфологической информации

На третьем этапе использовалась структурная информация, извлеченная на первом этапе (количество упоминаний, хэштегов, наличие ссылок), вместе с морфологической информацией второго этапа (наличие глаголов и имен собственных). Для объединения файлов с двумя наборами признаков был реализован алго-

ритм на языке программирования Python 3, программный код см. в приложении 5.

При одновременном использовании пяти признаков удалось правильно классифицировать 85% объектов тестовой выборки. В таб. 5 и 6 показаны данные о полноте и точности классификации и матрица ошибок классификации. Объединение признаков позволило увеличить результативность классификации относительно обоих предыдущих этапов.

Таблица 5. Точность и полнота классификации при использовании в качестве признаков информации о количестве упоминаний и хэштегов, а также о наличии ссылок, глаголов и имен собственных в текстах

	Точность	Полнота
Личные сообщения	0,858	0,930
Новости	0,832	0,693

Таблица 6. Матрица ошибок классификатора при использовании в качестве признаков информации о количестве упоминаний и хэштегов, а также о наличии ссылок, глаголов и имен собственных в текстах

Истинный класс	Классифицированы как	
	Личные сообщения	Новостные сообщения
Личные сообщения	278	21
Новостные сообщения	46	104

2.7. Анализ с помощью векторных представлений текстов

На четвертом этапе признаками служили необработанные тексты твитов, преобразованные программой Weka в векторный формат. При предварительной обработке данных (вкладка Preprosess) текстовые данные были по умолчанию восприняты как номинальные признаки. Чтобы преобразовать их в строчный формат, был использован фильтр weka.filters.unsupervised.attribute.NominaltoString.

Затем ко всем текстам выборки был применён фильтр weka.filters.unsupervised.attribute.StringToWordVector.

Параметры формирования векторов устанавливались в окне настройки фильтра. Окно настройки изображено на рисунке 6.

В ПО Weka предусмотрены разные способы наполнения словаря: его элементами могут быть последовательности символов, отделённые друг от друга разделителем из заданного набора (“WordTokenizer”), n-граммы – группы следующих друг за другом последовательностей символов (“NGramTokenizer”) и алфавитные символы (“AlphabeticTokenizer”).

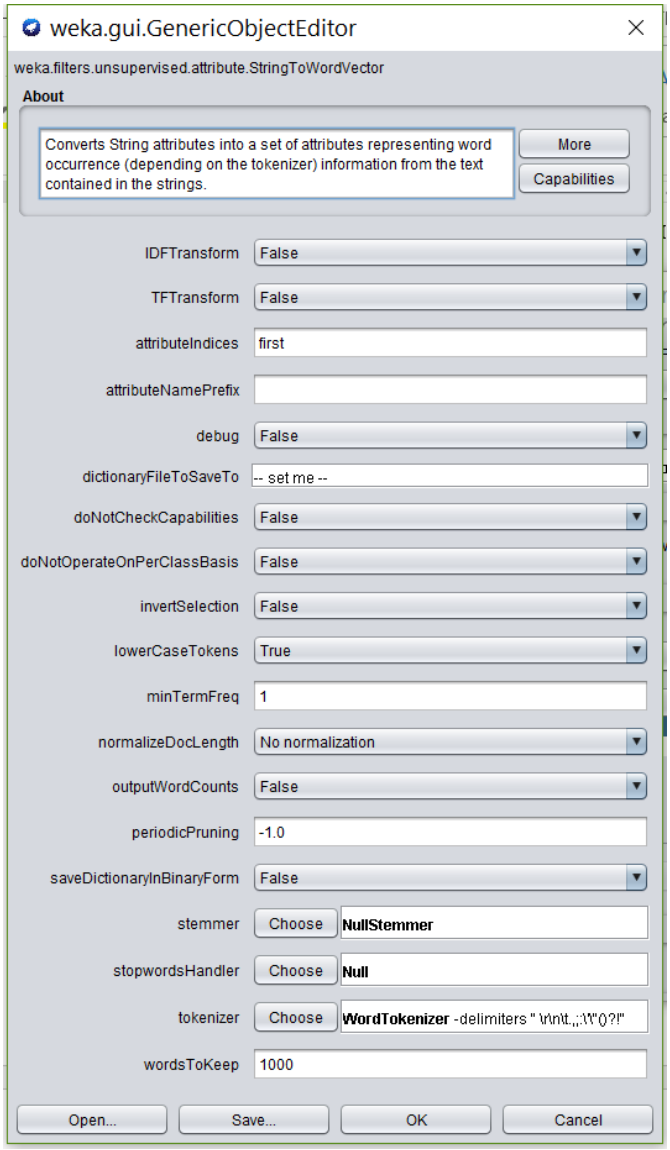

Рисунок 6. Окно настроек фильтра weka.filters.unsupervised.StringToWordVector

Для составления словаря на основе обучающей выборки были использованы алгоритмы WordTokenizer и NGramTokenizer (подсчитывались униграммы, биграммы и триграммы). Для работы с n-граммами из-за значительно возросшего количества признаков понадобилось увеличить объём выделяемой под работу программы памяти компьютера (этот параметр настраивается вручную в конфигурационном файле RunWeka.ini).

В зависимости от настроек фильтра StringToWordVector могут формироваться векторы, где каждый элемент отражает наличие или отсутствие данного терма в тексте (с параметром outputWordsCounts=False), векторы, где каждый элемент указывает на частоту вхождений данного терма в текст (outputWordsCounts=True), или векторы с весами TF-IDF – статистической мерой, используемой для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса (с параметрами IDFTransform=True и TFTransform=True) [Witten, Frank 2005].

В данной работе применялось формирование векторов на основе вхождений униграмм в текст. Мера TF-IDF и N-граммы не использовались, так как векторные представления текстов, сформированные с этими параметрами, не позволили улучшить результаты относительно векторов, сформированных на основе информации о наличии или отсутствии униграмм в тексте. Частотность слов не учитывалась, так как она малозначима в рамках очень коротких текстов, таких, как твиты.

При использовании векторов, учитывающих вхождение термов в текст, наивный байесовский классификатор верно классифицировал 402 из 449 объектов тестовой выборки, или 89,5%. Полнота выделения новостей выросла до 0,953, однако точность упала до 0,781. В таблицах 7 и 8 указана полнота и точность классификации обоих классов и количество верно и неверно классифицированных объектов обоих классов, соответственно.

Таблица 7. Точность и полнота классификации при использовании в качестве признаков векторных представлений необработанных текстов, учитывающих наличие или отсутствие терма в тексте

	Точность	Полнота
Личные сообщения	0,974	0,866
Новости	0,781	0,953

Таблица 8. Матрица ошибок классификатора при использовании в качестве признаков векторных представлений необработанных текстов, учитывающих наличие или отсутствие терма в тексте

Истинный класс	Классифицированы как	
	Личные сообщения	Новостные сообщения
Личные сообщения	259	40
Новостные сообщения	7	143

2.8. Анализ с помощью объединенного набора признаков

На заключительном этапе для лучшего формирования векторов на основе текстов твиты были лемматизированы. Weka позволяет использовать стеммеры при формировании векторов (LovinsStemmer, SnowballStemmer), но программа не предназначена для обработки текстов на русском языке, и в ней не заложены алгоритмы, показывающие хороший результат на материале русского языка.

Чтобы решить эту проблему, для лемматизации текстов, как и ранее для извлечения морфологической информации, использовался морфологический анализатор MyStem в оболочке для языка Python pymystem3.

На языке Python был реализован алгоритм, использующий модуль re стандартной библиотеки Python для удаления из текстов найденных в них ссылок на веб-страницы, знаков хэштега и упоминаний пользователей. Значения соответствующих признаков, а также данные о наличии в тексте глаголов и имен собственных записываются в файл вместе с остальной частью текста, лемматизированной при помощи модуля MyStem (программный код см. в приложении 6).

На данном этапе для классификации текстов использовались векторы, сформированные при помощи фильтра StringToWordVector (при построении векторных представлений учитывалось вхождение униграмм в текст) на основе лемматизированных текстов, вместе со структурной информацией (количество упоминаний и хэштегов, наличие ссылок) и морфологической информацией (наличие глаголов и имен собственных).

С использованием всех признаков наивный байесовский классификатор показал результат в 93,99%, или 422 верно классифицированных объекта из 449. В

таблице 9 указана полнота и точность классификации для двух классов текстов, в таблице 10 – количество верно и неверно классифицированных объектов двух классов.

Таблица 9. Точность и полнота классификации при одновременном использовании всех признаков

	Точность	Полнота
Личные сообщения	0,937	0,943
Новости	0,885	0,873

Таблица 10. Матрица ошибок классификатора при одновременном использовании всех признаков

Истинный класс	Классифицированы как	
	Личные сообщения	Новостные сообщения
Личные сообщения	282	17
Новостные сообщения	19	131

2.9. Сравнение результатов

Для сравнения результатов классификации на основе разных признаков были использованы данные о полноте и точности классификации, а также F-мера. F-мера – среднее гармоническое точности и полноты, эта величина позволяет одновременно учесть точность и полноту и рассчитывается по формуле:

$$F = 2 \cdot \frac{P \cdot R}{P + R},$$

где P – точность, R – полнота по классу.

При усложнении признаковых описаний удалось добиться последовательного улучшения результатов классификации. Использование только структурных признаков не дало удовлетворительных результатов, особенно низкой была полнота классификации новостей, составившая всего 0,213. F-мера по классу новостей составила 0,317.

Использование морфологических признаков позволило увеличить полноту выделения новостных сообщений до 0,647, а F-мера достигла 0,756.

Хотя результаты первой классификации были неудовлетворительными, объединение признаков из первого и второго наборов позволило улучшить совокупные результаты. F-мера по классу новостных текстов при третьей классификации была равна 0,756.

Использование векторных представлений текстов оказалось более эффективным. F-мера достигла 0,859 и 0,917 для новостных и личных текстов соответственно. Комбинация векторных представлений текстов со структурными и морфологическими признаками позволила увеличить эти показатели до 0,908 и 0,955 соответственно, при этом было верно классифицировано 93,99% тестовой выборки.

Величину F-меры по каждому классу на каждом из этапов см. на рисунке 7.

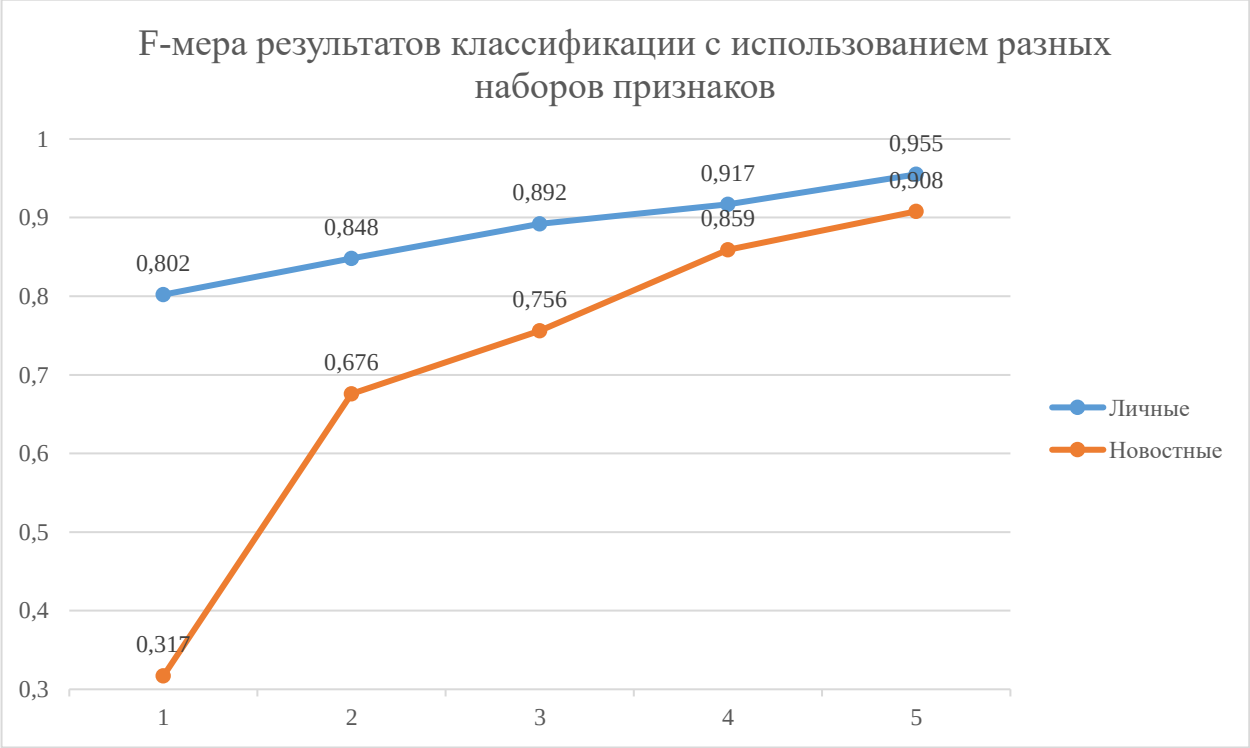

Рисунок 7. F-мера полноты и точности классификации личных и новостных текстов с использованием разных наборов признаков: 1 – количество упоминаний, хэштегов, наличие ссылок; 2 – наличие глаголов и имен собственных; 3 – упоминания, хэштеги, ссылки, глаголы, имена собственные; 4 – векторные представления текстов в исходном виде; 5 – векторные представления лемматизированных текстов и признаки из третьего набора

2.10. Выводы к главе 2

В этой главе описан эксперимент по выделению новостных сообщений с помощью машинного обучения. Наилучших результатов работы наивного байесов-

ского классификатора удалось достичь при одновременном использовании максимального количества признаков: структурной информации в текстах (упоминания, хэштеги, ссылки), морфологической информации (глаголы, имена собственные), а также векторных представлений очищенных от лишней информации и лемматизированных текстов. 93,99% твитов были верно классифицированы, полнота классификации личных и новостных сообщений составила 0,967 и 0,887 соответственно, точность – 0,944 и 0,930 соответственно.

При использовании каждого набора признаков F-мера по классу новостей была ниже, чем по классу личных сообщений. Это может быть обусловлено меньшим количеством новостных текстов в выборке, однако такое соотношение было выбрано намеренно, так как в случайной выборке непрерывно собираемых в Twitter сообщений новостей всегда значительно меньше, чем личных сообщений.

Результаты эксперимента показывают принципиальную возможность выделения новостных сообщений в Twitter при помощи вероятностного наивного байесовского классификатора и описанных наборов признаков.

Заключение

В работе был проведен эксперимент по выделению новостных текстов из массива русскоязычных сообщений в социальной сети Twitter с использованием методов машинного обучения.

Для исследования при помощи программы Webometric Analyst был собран массив твитов на русском языке. После предварительной обработки текстов была сформирована выборка из 1348 сообщений (898 сообщений личного характера и 450 новостей).

На языке программирования Python 3 были реализованы алгоритмы выделения различных наборов признаков текстов. На разных этапах было использовано 5 наборов признаков:

1. упоминания пользователей, хэштеги и ссылки в твитах;
2. глаголы и имена собственные в твитах;
3. упоминания, хэштеги, ссылки + глаголы, имена собственные;
4. тексты твитов, преобразованные в векторный формат;
5. тексты твитов, очищенные от ссылок и упоминаний, лемматизированные при помощи морфологического анализатора MyStem и затем преобразованные в векторный формат, вместе с информацией об упоминаниях, хэштегах, ссылках, глаголах и именах собственных в исходном тексте твита.

Программа Weka была настроена для работы с русскоязычными текстами (рабочая кодировка изменена на UTF-8, увеличен объем выделяемой на работу программы памяти). Была сформирована обучающая выборка из 899 твитов (599 личных и 300 новостных текстов) и тестовая выборка из 449 твитов (299 личных и 150 новостных текстов). При загрузке в программу Weka типы данных были приведены к нужным при помощи встроенных фильтров, все данные были сохранены в формате .arff. Для классификации был применен наивный байесовский классификатор. Модель проходила обучение на обучающей выборке, эффективность оценивалась по результатам классификации тестовой выборки.

Полнота и точность классификации последовательно улучшалась при использовании более сложных комбинаций признаков. Классификация по количеству упоминаний и хэштегов и наличию ссылок в тексте показала результат в 69,3% верно классифицированных объектов тестовой выборки, но полнота классификации новостных твитов составила лишь 0,213, т.е. для распознавания новостей этих признаков было недостаточно. Использование информации о наличии глаголов и имен собственных позволило увеличить полноту классификации новостей до 0,647, а одновременное использование пяти признаков – до 0,693, причем F-мера стала равна 0,756, и 85% объектов тестовой выборки было верно классифицировано. При классификации по векторным представлениям текстов в исходном виде F-мера полноты и точности для личных и новостных твитов составила 0,917 и 0,859 соответственно. После удаления из текстов лишней информации, лемматизации и добавления признаков из первого и второго набора удалось достичь увеличения F-меры для личных и новостных твитов до 0,955 и 0,908 соответственно. Верно классифицированы были 93,99% объектов тестовой выборки.

Эти результаты позволяют сделать вывод, что алгоритмы машинного обучения и, в частности, наивный байесовский классификатор могут быть с успехом использованы для выделения новостей из массива твитов на русском языке. Одновременное использование различных типов признаков позволяет достичь наилучших результатов.

Список литературы

1. Адаскина Ю.В., Паничева П.В., Попов А.М. Использование синтаксиса для анализа тональности твитов на русском языке // Электронные библиотеки. – Казань, 2015. – Т. 18, № 3-4. – С. 163-184. – [Электронный ресурс]. URL: http://ojs.kpfu.ru/index.php/elbib/article/view/21/11 (дата обращения: 18.05.2017).

2. Батура Т.В. Математическая лингвистика и автоматическая обработка текстов на естественном языке: учебное пособие. – Новосибирск: РИЦ НГУ, 2016. – 166 с.

3. Васильев В.Г., Худякова М.В., Давыдов С. Классификация отзывов пользователей с использованием фрагментных правил // Компьютерная лингвистика и интеллектуальные технологии. – М., 2012. – Вып. 11, т. 2. – С. 66-76. – [Электронный ресурс]. URL: http://www.dialog-21.ru/media/1384/132.pdf (дата обращения: 18.05.2017).

4. Воронцов К.В. Математические методы обучения по прецедентам (теория обучения машин). – 2007. – [Электронный ресурс]. URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf (дата обращения: 18.05.2017).

5. Епрев А.С. Автоматическая классификация текстовых документов // Математические структуры и моделирование / Под ред. А.К. Гуца. – Омск: "Омское книжное издательство", 2010. – Вып. 21. – С. 65-81. – [Электронный ресурс]. URL: http://msm.univer.omsk.su/sbornik/jrn21/sbornik_n21.pdf (дата обращения: 18.05.2017).

6. Котельников Е.В., Клековкина М.В. Автоматический анализ тональности текстов на основе методов машинного обучения // Компьютерная лингвистика и интеллектуальные технологии. – М., 2012. – Вып. 11, т. 2. – С. 27-36. – [Электронный ресурс]. URL: http://www.dialog-21.ru/media/1380/105.pdf (дата обращения: 18.05.2017).

7. Найденова К.А., Невзорова О.А. Машинное обучение в задачах обработки естественного языка: обзор современного состояния исследований // Ученые

записки Казанского государственного университета. – Казань, 2008. – Том 150, кн. 4. – С. 5-24.

8. Протопопова Е.В., Букия Г.Т. Машинное обучение в лингвистике // Прикладная и компьютерная лингвистика / Николаев И.С., Митренина О.В., Ландо Т.М. (ред.). – М.: УРСС, 2016. – 320 с.

9. Расшифровка граммем // Документация MyStem. – [Электронный ресурс]. URL: https://tech.yandex.ru/mystem/doc/grammemes-values-docpage/ (дата обращения: 18.05.2017).

10. Романов А.С., Мещеряков Р.В. Определение пола автора короткого электронного сообщения // Компьютерная лингвистика и интеллектуальные технологии. – М., 2011. – Вып. 10. – С. 556-561. – [Электронный ресурс]. URL: http://www.dialog-21.ru/media/1456/55.pdf (дата обращения: 18.05.2017).

11. Chikersal P., Poria S., Cambria E., Gelbukh A., Siong C.E. Modelling Public Sentiment in Twitter: Using Linguistic Patterns to Enhance Supervised Learning // Computational Linguistics and Intelligent Text Processing. Part II. – Springer, 2015. – P. 49-65.

12. Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., Witten I.H. The WEKA Data Mining Software: An Update // SIGKDD Explorations Newsletter. – 2009. – Volume 11, Issue 1. – P. 10-18. – [Электронный ресурс]. URL: http://www.cms.waikato.ac.nz/~ml/publications/2009/weka_update.pdf (дата обращения: 18.05.2017).

13. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. – Springer, 2009. – 745 p.

14. Jindal N., Liu B. Opinion Spam and Analysis // Proceedings of the 2008 International Conference on Web Search and Data Mining. – Palo Alto, 2008. – [Электронный ресурс]. URL: https://pdfs.semanticscholar.org/16f2/deb863ef6d3d6f432de12a2e81149ab03e5a.pdf (дата обращения: 18.05.2017).

15. Kouloumpis E., Wilson T., Moore J.D. Twitter Sentiment Analysis: The Good the Bad and the OMG! // Proceedings of the Fifth International Conference on Weblogs and

Social Media. – AAAI Press, 2011. – P. 538-541. – [Электронный ресурс]. URL: http://www.research.ed.ac.uk/portal/files/18447136/Kouloumpis_Wilson_ET_AL_2011_Twitter_Sentiment_Analysis_The_Good_the_Bad_and_the_OMG.pdf (дата обращения: 18.05.2017).

16. Kunneman F., Bosch A. van den. Event detection in Twitter: A machine-learning approach based on term pivoting // Proceedings of the 26th Benelux Conference on Artificial Intelligence / Grootjen, F., Otworowska, M., Kwisthout, J. (ed.). – Nijmegen, 2014. – P. 65-72. – [Электронный ресурс]. URL: http://antalvandenbosch.ruhosting.nl/papers/event-detection-twitter.pdf (дата обращения: 18.05.2017).

17. Lai M., Farías D.I.H., Patti V., Rosso P. Friends and Enemies of Clinton and Trump: Using Context for Detecting Stance in Political Tweets // Proceedings of the 15th Mexican International Conference on Artificial Intelligence. – Cancún, 2016. – [Электронный ресурс]. URL: https://arxiv.org/pdf/1702.08021.pdf (дата обращения: 18.05.2017).

18. Li H., Mukherjee A., Liu B., Kornfield R., Emery S. Detecting Campaign Promoters on Twitter using Markov Random Fields // Proceedings of the 2014 IEEE International Conference on Data Mining. – Washington, DC, 2014. – P. 290-299. – [Электронный ресурс]. URL: https://www.cs.uic.edu/~liub/publications/twitter-promoters-paper531.pdf (дата обращения: 18.05.2017).

19. McCallum A., Nigam K. A Comparison of Event Models for Naive Bayes Text Classification // AAAI/ICML-98 Workshop on Learning for Text Categorization. – Madison, 1998. – P. 41-48. – [Электронный ресурс]. URL: http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf (дата обращения: 18.05.2017).

20. Melville P., Gryc W., Lawrence R.D. Sentiment Analysis of Blogs by Combining Lexical Knowledge with Text Classification // Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. – Paris, 2009. – P. 1275-1284. – [Электронный ресурс]. URL: http://www.prem-melville.com/publications/pooling-multinomials-kdd09.pdf (дата обращения:

18.05.2017).

21. Melville P., Sindhwani V., Lawrence R.D., Meliksetian E., Liu Y., Hsueh P.-Y., Perlich C. Machine Learning for Social Media Analytics // Machine Learning Symposium, New York Academy of Sciences. – New York, 2009. – [Электронный ресурс]. URL: http://www.prem-melville.com/publications/sma-nyas09.pdf (дата обращения: 18.05.2017).

22. Mitchell T. Machine Learning. – McGraw-Hill Science/Engineering/Math, 1997. – 432 p.

23. Mukherjee A., Liu B. Improving Gender Classification of Blog Authors // Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. – Cambridge, Massachusetts, 2010. – P. 207-217. – [Электронный ресурс]. URL: http://www.aclweb.org/anthology/D10-1021 (дата обращения: 18.05.2017).

24. Pennacchiotti M., Popescu A.-M. A Machine Learning Approach to Twitter User Classification // Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media. – Barcelona, 2011. – P. 281-288. – [Электронный ресурс]. URL: https://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2886/3262 (дата обращения: 18.05.2017).

25. Rakhilina E., Vyrenkova A., Mustakimova E., Ladygina A., Smirnov I. Building a learner corpus for Russian // Proceedings of the joint workshop on NLP for Computer Assisted Language Learning and NLP for Language Acquisition. – Umeå, 2016. – P. 66-75. – [Электронный ресурс]. URL: http://www.ep.liu.se/ecp/130/ecp16130.pdf (дата обращения: 18.05.2017).

26. Rish I. An empirical study of the naive Bayes classifier // IJCAI 2001 workshop on empirical methods in artificial intelligence. – IBM New York, 2001. – Vol. 3, Issue 22. – P. 41-46. – [Электронный ресурс]. URL: http://www.research.ibm.com/people/r/rish/papers/RC22230.pdf (дата обращения: 18.05.2017).

27. Sebastiani F. Machine Learning in Automated Text Categorization // ACM Computing Surveys (CSUR). – New York, 2002. – Vol. 34, No. 1. – P. 1-47. –

[Электронный ресурс]. URL: http://nmis.isti.cnr.it/sebastiani/Publications/ACMCS02.pdf (дата обращения: 18.05.2017).

28. Segalovich I. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine // Proceedings of the International Conference on Machine Learning: Models, Technologies and Applications. – Las Vegas, 2003. – [Электронный ресурс]. URL: http://cache-spb03.cdn.yandex.net/download.yandex.ru/company/iseg-las-vegas.pdf (дата обращения: 18.05.2017).

29. Thelwall M. Introduction to Webometrics: Quantitative Web Research for the Social Sciences. – Morgan & Claypool Publishers, 2009. – 126 p.

30. Vikre L.C., Wold H.M. Online News Detection on Twitter. – Norwegian University of Science and Technology, 2015. – 87 p.

31. Wan Y., Gao Q. An Ensemble Sentiment Classification System of Twitter Data for Airline Services Analysis // Proceedings of 15th IEEE International Conference on Data Mining Workshop – 2015. – [Электронный ресурс]. URL: http://sentic.net/sentire2015wan.pdf (дата обращения: 14.05.2017).

32. Witten I.H., Frank E. Data mining. Practical machine learning tools and techniques, Second Edition. – Elsevier, 2005. – 525 p.

Приложения

Приложение 1. Примеры новостных текстов

"Крупное ДТП в Северной Осетии: автобус, следовавший из Москвы в Ереван, столкнулся с автовозом https://t.co/9ZvyL67Lde"

"Лиговский в дыму из-за пожара в заброшенном доме https://t.co/fCpahnMMoh"

"Операция по освобождению Алеппо вступила в завершающую стадию #Сирия #Асад #боевики #саперы #новости #news https://t.co/QIfFVhghfE"

"Ильхам Алиев назначил первым вице-президентом Азербайджана свою жену https://t.co/5zyfgXxPyH #алиев #азербайджан… https://t.co/Y5sy9vWH5G"

"Родители ребенка, эвакуированного вместе с машиной в Иркутске, забрали его со штрафстоянки. А отцу выписали штраф за неправильную парковку "

"Из-за сильного ветра в Петербурге частично закрыли дамбу https://t.co/rXtqfrk2Mx"

"Грабитель с молотком унес «ювелирку» из магазина на Звездной https://t.co/WqaPhGcLRu"

"В США прошел второй раунд теледебатов кандидатов в президенты https://t.co/WrHR52CQjl"

"В Греции проходит общенациональная #забастовка #Греция #ЕС #реформы #налоги #пенсии #новости #news https://t.co/4tv2XDT1Us"

"Информационный стенд у «Авроры» поменяют из-за орфографических ошибок https://t.co/7klYS47Sfa https://t.co/RS6kipxc61"

"На Московском центральном кольце открываются две новые станции https://t.co/KL2Fhoy6Pv"

"Найден неизвестный роман одного из самых знаменитых американских писателей XIX века https://t.co/uMb2g1Tqzy https://t.co/C99oXWa1aL"

"Восемнадцать тонн мандаринов украли в Петербурге https://t.co/0dqFOqsdxS"

Приложение 2. Примеры текстов личного характера

"какой вид транспорта при передвижении на дальние расстояния вы предпочитаете и какой считаете самым безопасным?"

"Как переключить соседскую собаку в режим без звука? #вопрос"

"Как вы думаете, девушке пойдет пирсинг брови или это слишком брутально?"

"При проблемах с женщинами надо поступать исключительно так, как подобает настоящему мужчине. Убегать."

"До зачета день я написала 8 карточек из 21"

"@patsko200015 Ахах, нее успокойся!)"

"Сижу допиваю шампанское, в 4 утра вставать :("

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"@bolshe4em @oz_6454 Просто мы привыкли,что реперы похожи на сусликов.Так оно и есть."

"Сладких снов ,ребята :) ? Завтра будет новый день (когда ты кэп) удачки всем https://t.co/AT4z6ioGcB"

"RT @love___mommy: ЛЕНТАААА ЭТО НАШ НОВЫЙ УЧИТЕЛЬ ПО ГЕОГРАФИИ https://t.co/DriFyjhG79"

"Как вы подрабатываете, если не хватает денег?"

"@alina_sanaeva а мне не на кого ходить :с на русских"

"одно и тоже изо дня в день #рутина"

"жрать без мяса грустно, но можно"

"@Anatoly40A а я на это еще и в кино ходила :D"

"Отдам маленький кусочик счастья в хорошие руки кушаем к лотку преучены сибирской породы имеем короткий хвостик"

"Я человек простой, если таксист не пытался заговорить со мной во время поездки, ставлю 5 за обслуживание"

"Много пишет ребят,которые предлагают пройти курсы пикапа, я что так плох что ли? :("

Приложение 3. Алгоритм выделения количества упоминаний и хэштегов, наличия ссылок в текстах

```
# -*- coding: utf-8 -*-
"""
@author: LizaK
"""

import re

def WriteCsv(inputfile,outputfile):
    """
    Принимает путь к файлу с текстами и метками классов и к файлу для записи
    результатов, создает файл, в который записывает строку с названиями
    признаков и значения признаков (количество упоминаний, хэштегов,
    наличие ссылок в тексте) для каждого объекта, разделенные запятыми,
    и метку класса объекта
    """
    linecount = LineCount(inputfile)
    with open(outputfile, 'w', encoding='utf-8') as writefile:
        #Записываем шапку файла
        writefile.write("Mentions,Hashtags,Links,Label\n")
        with open(inputfile, 'r', encoding='utf-8') as readfile:
            for i in range(linecount):
                line = readfile.readline()
                dicti = WriteDictionary(line)
                #Записываем строки
                if i!=0:
                    writefile.write(str(dicti["mentions"])+','+
                    str(dicti["hashtags"])+','+
                    str(dicti["links"])+','+str(dicti["label"]+'\n'))

def LineCount(file):
    """
    Принимает путь к файлу, возвращает количество строк в нем
    """
    return(len(open(file, 'r', encoding='utf-8').readlines()))

def WriteDictionary(line):
    """
    Принимает строку формата '"text",label', возвращает словарь, где ключи -
    имена полей (mentions, hashtags, links, label), значения - соответсвующие
    значения для строк
    """
    text = line[line.find('"')+1:-4]
    label = line[-2]
    onedict = {}
    onedict.setdefault("text", text)
    onedict.setdefault("label", label)
    onedict.setdefault("mentions", FindMentions(text))
    onedict.setdefault("hashtags", FindHashtags(text))
    onedict.setdefault("links", FindLinks(text))
    return(onedict)

def FindMentions(text):
    """
```

```
    Принимает строку, возвращает количество упоминаний пользователей (@xxx)
    в ней
    """
    return(len(re.findall(r"@[A-Za-z0-9_]+", text)))

def FindHashtags(text):
    """
    Принимает строку, возвращает количество хештэгов (#xxx) в ней
    """
    return(len(re.findall(r"#\w+", text)))

def FindLinks(text):
    """
    Принимает строку, возвращает 1, если в ней есть ссылка (httpxxx), 0,
    если ссылок нет
    """
    if len(re.findall(r"http[A-Za-z0-9_\-.:/]+", text)) != 0:
        return(1)
    else:
        return(0)
```

Приложение 4. Алгоритм выделения текстов, содержащих глаголы и имена собственные

```
# -*- coding: utf-8 -*-
"""
@author: LizaK
"""

import re
from pymystem3 import Mystem

def WriteVerbsAndNames(inputfile,outputfile):
    """
    Принимает путь к файлу с текстами и метками классов и к файлу для записи
    результатов, создает файл, в который записывает строку с названиями
    признаков и значения признаков (наличие глаголов и имен собственных
    в тексте) для каждого объекта, разделенные запятыми, и метку класса объекта
    """
    linecount = LineCount(inputfile)
    with open(outputfile, 'w', encoding='utf-8') as writefile:
        #Записываем шапку файла
        writefile.write("Verbs,Names,Label\n")
        with open(inputfile, 'r', encoding='utf-8') as readfile:
            for i in range(linecount):
                line = readfile.readline()
                dicti = WriteVNDictionary(line)
                #Записываем строки
                if i!=0:
                    writefile.write(str(dicti["verbs"])+','+
                    str(dicti["names"])+','+str(dicti["label"]+'\n'))

def LineCount(file):
    """
    Принимает путь к файлу, возвращает количество строк в нем
    """
    return(len(open(file, 'r', encoding='utf-8').readlines()))

def WriteVNDictionary(line):
    """
    Принимает строку формата '"text",label', возвращает словарь, где ключи -
    имена полей (verbs, names, label), значения - соответствующие значения
    для строк
    """
    text = line[line.find('"')+1:-4]
    label = line[-2]
    onedict = {}
    onedict.setdefault("text", text)
    onedict.setdefault("label", label)
    onedict.setdefault("verbs", FindVerbs(text))
    onedict.setdefault("names", FindNames(text))
    return(onedict)

def FindVerbs(text):
    """
    Принимает строку, возвращает 1, если MyStem определил хотя бы одну
    словоформу в строке как глагол, 0, если нет
```

```
    """
    text = Clean(text)
    analyzer = Mystem()
    textinfo = analyzer.analyze(text)
    for wordinfo in textinfo:
        if "analysis" not in wordinfo.keys():
            continue
        elif wordinfo["analysis"] == []:
            continue
        elif len(wordinfo["analysis"][0]["gr"]) < 1:
            continue
        else:
            if wordinfo["analysis"][0]["gr"][0] == "V":
                return 1
                break
            else:
                continue
    return 0

def FindNames(text):
    """
    Принимает строку, возвращает 1, если MyStem определил хотя бы одну
    словоформу в строке как географическое название, имя или фамилию,
    и при этом эта словоформа не помечена как ошибочная, 0, если нет
    """
    text = Clean(text)
    analyzer = Mystem()
    textinfo = analyzer.analyze(text)
    for wordinfo in textinfo:
        if "analysis" not in wordinfo.keys():
            continue
        elif wordinfo["analysis"] == []:
            continue
        elif ("qual" in wordinfo["analysis"][0].keys() and
        wordinfo["analysis"][0]["qual"] == "bastard"):
            continue
        elif len(wordinfo["analysis"][0]["gr"]) < 5:
            continue
        else:
            if (wordinfo["analysis"][0]["gr"][2:5] == "гео" or
            wordinfo["analysis"][0]["gr"][2:5] == "имя" or
            wordinfo["analysis"][0]["gr"][2:5] == "фам"):
                return 1
                break
            else:
                continue
    return 0

def Clean(text):
    """
    Принимает строку, удаляет из нее упоминания, символы хештэга и ссылки
    """
    mentions = re.compile(r"@[A-Za-z0-9_]+")
    for mention in mentions.finditer(text):
        text = text.replace(mention.group(), '')
    hashtags = re.compile(r"#\w+")
    for hashtag in hashtags.finditer(text):
```

```
    text = text.replace(hashtag.group(), hashtag.group()[1:])
links = re.compile(r"http[A-Za-z0-9_\-.:/]+")
for link in links.finditer(text):
    text = text.replace(link.group(), '')
return(text)
```

Приложение 5. Алгоритм объединения файлов с признаками, использованными на первом и втором этапах анализа

```
# -*- coding: utf-8 -*-
"""
@author: LizaK
"""

def WriteMHLandVN(MHLfile,VNfile,outputfile):
    """
    Принимает путь к файлу со значениями структурных признаков, к файлу
    со значениями морфологических признаков, к файлу для записи,
    записывает в файл для записи все значения признаков и метки классов
    """
    #Проверяем, одинаковой ли длины файлы
    if LineCount(MHLfile) != LineCount(VNfile):
        return "INCOMPATIBLE FILES"
    linecount = LineCount(MHLfile)
    with open(outputfile, 'w', encoding='utf-8') as finalfile:
        #Записываем шапку файла
        finalfile.write("Mentions,Hashtags,Links,Verbs,Names,Label\n")
        with open(MHLfile, 'r', encoding='utf-8') as mhlfile:
            with open(VNfile, 'r', encoding='utf-8') as vnfile:
                for i in range(linecount):
                    mhlline = mhlfile.readline()
                    vnline = vnfile.readline()
                    #Записываем строки, соответствующие объектам
                    if i!=0:
                        finalfile.write(mhlline.split(',')[0]+','+
                        mhlline.split(',')[1]+','+
                        mhlline.split(',')[2]+','+vnline)

def LineCount(file):
    """
    Принимает путь к файлу, возвращает количество строк в нем
    """
    return(len(open(file, 'r', encoding='utf-8').readlines()))
```

Приложение 6. Алгоритм удаления ссылок, имен пользователей и символов хэштега из текстов, записи в файл лемматизированных текстов и объединенного набора признаков

```
# -*- coding: utf-8 -*-
"""
@author: LizaK
"""

from pymystem3 import Mystem
import re

def WriteAllFeatures(inputfile,outputfile):
    """
    Принимает путь к файлу с текстами и метками классов и к файлу для записи
    результатов, создает файл, в который записывает строку с названиями
    признаков, лемматизированные и очищенные тексты твитов, значения признаков
    (упоминания, хэштеги, ссылки, глаголы, имена собственные в тексте)
    для каждого объекта, разделенные запятыми, и метку класса объекта
    """
    linecount = LineCount(inputfile)
    with open(outputfile, 'w', encoding='utf-8') as writefile:
        #Записываем шапку
        writefile.write("LemText,Mentions,Hashtags,Links,Verbs,Names,Label\n")
        with open(inputfile, 'r', encoding='utf-8') as readfile:
            for i in range(linecount):
                line = readfile.readline()
                dicti = WriteAllFeaturesDictionary(line)
                #Записываем строки
                if i!=0:
                    writefile.write('"'+CleanLemmas(dicti["text"])+'",'+
                    str(dicti["mentions"])+','+str(dicti["hashtags"])+','+
                    str(dicti["links"])+','+str(dicti["verbs"])+','+
                    str(dicti["names"])+','+str(dicti["label"])+'\n')

def LineCount(file):
    """
    Принимает путь к файлу, возвращает количество строк в нем
    """
    return(len(open(file, 'r', encoding='utf-8').readlines()))

def WriteAllFeaturesDictionary(line):
    """
    Принимает строку формата '"text",label', возвращает словарь, где ключи -
    имена полей (mentions, hashtags, links, verbs, names, label), значения -
    соответсвующие значения для строк
    """
    text = line[line.find('"')+1:-4]
    label = line[-2]
    onedict = {}
    onedict.setdefault("text", text)
    onedict.setdefault("label", label)
    onedict.setdefault("mentions", FindMentions(text))
    onedict.setdefault("hashtags", FindHashtags(text))
    onedict.setdefault("links", FindLinks(text))
```

```
    onedict.setdefault("verbs", FindVerbs(text))
    onedict.setdefault("names", FindNames(text))
    return(onedict)


def CleanLemmas(text):
    """
    Принимает строку, лемматизирует с помощью MyStem (ф-я Stem), удаляет
    из списка лемм пунктуацию и специальные символы, возвращает строку
    из оставшихся лемм, разделенных пробелами
    """
    lemmas = Stem(text)[0:-1]
    punct = re.compile(r"[…!'"'$%&"'"()*+,./:;<=>?@\^_`{|}~\- ]")
    for lemma in lemmas:
        if punct.match(lemma) != None:
            lemmas.remove(lemma)
    return(' '.join(lemmas))

def Stem(text):
    """
    Принимает строку, удаляет из нее упоминания, символы хештэга и ссылки
    (ф-я Clean), лемматизирует при помощи MyStem, возвращает список лемм
    """
    text = Clean(text)
    lemmatizer = Mystem()
    lemmas = lemmatizer.lemmatize(text)
    return(lemmas)

def Clean(text):
    """
    Принимает строку, удаляет из нее упоминания, символы хештэга и ссылки
    """
    mentions = re.compile(r"@[A-Za-z0-9_]+")
    for mention in mentions.finditer(text):
        text = text.replace(mention.group(), '')
    hashtags = re.compile(r"#\w+")
    for hashtag in hashtags.finditer(text):
        text = text.replace(hashtag.group(), hashtag.group()[1:])
    links = re.compile(r"http[A-Za-z0-9_\-.:/]+")
    for link in links.finditer(text):
        text = text.replace(link.group(), '')
    return(text)

def FindMentions(text):
    """
    Принимает строку, возвращает количество упоминаний пользователей (@xxx)
    в ней
    """
    return(len(re.findall(r"@[A-Za-z0-9_]+", text)))

def FindHashtags(text):
    """
    Принимает строку, возвращает количество хештэгов (#xxx) в ней
    """
    return(len(re.findall(r"#\w+", text)))

def FindLinks(text):
```

```
    """
    Принимает строку, возвращает 1, если в ней есть ссылка (httpxxx), 0,
    если ссылок нет
    """
    if len(re.findall(r"http[A-Za-z0-9_\-.:/]+", text)) != 0:
        return(1)
    else:
        return(0)

def FindVerbs(text):
    """
    Принимает строку, возвращает 1, если MyStem определил хотя бы одну
    словоформу в строке как глагол, 0, если нет
    """
    text = Clean(text)
    analyzer = Mystem()
    textinfo = analyzer.analyze(text)
    for wordinfo in textinfo:
        if "analysis" not in wordinfo.keys():
            continue
        elif wordinfo["analysis"] == []:
            continue
        elif len(wordinfo["analysis"][0]["gr"]) < 1:
            continue
        else:
            if wordinfo["analysis"][0]["gr"][0] == "V":
                return 1
                break
            else:
                continue
    return 0

def FindNames(text):
    """
    Принимает строку, возвращает 1, если MyStem определил хотя бы одну
    словоформу в строке как географическое название, имя или фамилию,
    и при этом эта словоформа не помечена как ошибочная, 0, если нет
    """
    text = Clean(text)
    analyzer = Mystem()
    textinfo = analyzer.analyze(text)
    for wordinfo in textinfo:
        if "analysis" not in wordinfo.keys():
            continue
        elif wordinfo["analysis"] == []:
            continue
        elif ("qual" in wordinfo["analysis"][0].keys() and
        wordinfo["analysis"][0]["qual"] == "bastard"):
            continue
        elif len(wordinfo["analysis"][0]["gr"]) < 5:
            continue
        else:
            if (wordinfo["analysis"][0]["gr"][2:5] == "гео" or
            wordinfo["analysis"][0]["gr"][2:5] == "имя" or
            wordinfo["analysis"][0]["gr"][2:5] == "фам"):
                return 1
                break
```

```
        else:
            continue
return 0
```